There is No Free Lunch: Tradeoffs in the Utility of Learned Knowledge

SMADAR KEDAR
KATHLEEN MCKUSICK
ARTIFICIAL INTELLIGENCE RESEARCH BRANCH
MS 269-2
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035-1000

AN EXTENDED ABSTRACT OF THIS PAPER IS PUBLISHED IN THE "PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE PLANNING SYSTEMS", JAMES HENDLER (ED.), SILVER SPRINGS, MD., JUNE, 1992

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-92-04

June, 1992

There is no Free Lunch: Tradeoffs in the Utility of Learned Knowledge

Smadar T. Kedar
Kathleen B. McKusick
NASA Ames Research Center
Artificial Intelligence Research Branch
Mail Stop 269-2
Moffett Field, CA 94035

kedar@ptolemy.arc.nasa.gov
mckusick@ptolemy.arc.nasa.gov

June 12, 1992

Abstract

With the recent introduction of learning in integrated systems, there is a need to measure the utility of learned knowledge for these more complex systems. A difficulty arrises when there are multiple, possibly conflicting, utility metrics to be measured. In this paper, we present schemes which trade off conflicting utility metrics in order to achieve some global performance objectives. In particular, we present a case study of a multi-strategy machine learning system, mutual theory refinement, which refines world models for an integrated reactive system, the Entropy Reduction Engine. We provide experimental results on the utility of learned knowledge in two conflicting metrics – improved accuracy and degraded efficiency. We then demonstrate two ways to trade off these metrics. In each, some learned knowledge is either approximated or dynamically 'forgotten' so as to improve efficiency while degrading accuracy only slightly.

- **Function:** Planning, learning
- **Other keywords:** Reactivity, theory refinement, failure-driven learning

1 Introduction and Motivation

Successful planning and control systems in realistic domains depend on the ability to improve with experience. One characteristic of such systems is the ability to gracefully recover from failures, and avoid similar failures in the future. The long term objective of our machine learning research (Kedar *et. al.*, 1991; Kedar & Chrisman, 1992) is to improve systems for planning and control by autonomously and systematically detecting failures, and refining domain knowledge so as to avoid such failures in the future.

Although for many years machine learning research has defined its primary goal as performance system improvement (Simon, 1983) , it is only in the past five years or so that the field has sufficiently matured to facilitate experimental validation of the *utility* of learned knowledge[1] for performance (Langley & Kibler, 1991). The notion of a *utility problem* was first presented in (Minton, 1988), referring to the degradation of system performance by machine learning. Minton shows that efficiency degrades, rather than improves, when certain search control rules are acquired using explanation-based learning. This results from learned rules whose high match cost overrides benefits in reducing search cost. Holder defines the *general utility problem* as the "degradation of performance due to increasing amounts of learned knowledge" (Holder, 1988). This notion generalizes utility to other learning paradigms and other performance metrics. For example, degraded *efficiency* due to increased match cost caused by analytic learning is similar to the problem of decreased *accuracy* in overfitting caused by empirical learning. In a parallel effort, the AI planning community has taken a recent interest in experimentally validating its research, attempting to identify what constitutes a successful planning system through more rigorous experimentation and analysis (Langley & Drummond 1991), (Al-Badr & Hanks, 1991).

Despite these contributions, research on utility is still myopic. Most of the approaches to utility analysis focus on a single performance system, a single learning paradigm, and a single measure of utility (e.g. efficiency in Minton, 1988; Tambe 1990; or accuracy in Holder, 1991). Yet with the recent introduction of learning in integrated systems, there is a need to measure

[1]Because of unfortunate choice in terminology, the 'utility' of learned knowledge as referred to in machine learning is not technically related to the decision-theoretic concept. 'Utility' in decision theory is a value or cost of a state (see (Berger, 1980) for an introduction to decision theory).

the utility of learned knowledge for these more complex systems. A difficulty arises when there are multiple, possibly conflicting, utility metrics to be measured.

In this paper, we present schemes which trade off conflicting utility metrics in order to achieve some global performance objectives. In particular, we present a case study of a multi-strategy machine learning system, *mutual theory refinement*, which refines world models for an integrated reactive system, the Entropy Reduction Engine (Bresina & Drummond, 1990). We provide experimental results on the utility of learned knowledge in two conflicting metrics – improved accuracy and degraded efficiency. We then demonstrate methods of trading off these metrics, whereby some learned knowledge is approximated so as to improve efficiency while degrading accuracy only slightly.

In Section 2 we familiarize the reader with the architecture of the case study. Next, in Section 3, we describe our experimental results on the utility of the learned knowledge for the case study. In Section 4, we step back from the details of the particular case study to discuss the general issue of tradeoffs of different utility measures for multi-strategy learning in integrated systems. We present our conclusions in Section 5.

2 The Architecture

In this section we describe the architectures of the performance and learning systems. We focus only on those aspects directly relevant to our experimentation (see associated references for a more thorough treatment). The current testbed for the integrated system discussed below is the NASA TileWorld experimental domain (Philips, 1991). This TileWorld consists of a simulator of a single agent in a two-dimensional grid of cells, able to move in four compass directions, grasp and release tiles. The domain is simple, yet has challenging aspects of real-world domains such as actions with stochastic outcomes, and unexpected external events such as winds[2].

2.1 The Entropy Reduction Engine System

Reactive systems are situated in an environment in which they sense and act. Our case study is cast within one such system, the Entropy Reduction Engine (ERE) (Bresina & Drummond, 1990; Drummond, Bresina, & Kedar, 1991). Unlike systems which consist of hand-coded reactions tailored to a particular task

[2]For a discussion as to the usefulness of the tileworld as a domain for experimentation, see (Pollack, 1990; Al-Badr & Hanks 1991).

(notably Brooks's (1986) subsumption architecture), ERE integrates planning and scheduling with reaction in order to *automatically* synthesize reactions appropriate to the given goals and environment. For the purposes of the paper, we present a high-level algorithmic description of two components of ERE: the planner and the reactor. Given as input a set of operators and constraints on the domain, the *planner* performs *temporal projection* (a version of forward search) to find a sequence of operators to achieve the goal. Using goal regression, the plan is then compiled into a set of *situated control rules* (SCR's). The SCR's provide 'advice' from the planner to the *reactor* as to which operator to apply when, so as to achieve the goal. The reactor either executes an applicable SCR, or resorts to an applicable random action.

ERE models the world with operators and domain constraints. The operators model the agent's actions in terms of preconditions and effects. The operators are further divided into *specializations* containing additional preconditions and specialized effects. Operator effects are a set of nondeterministic *variant outcomes*, with associated probabilities of occurrence. *Domain constraints* model physical laws, e.g., "the agent cannot be in two locations at once." Actions are performed in the world (or simulator), which moves from one state to the next. The operators and constraints are only approximate models, and therefore may not always predict the observed results of actions. That may lead to prediction failures, which drive the learning system to refine these world models.

2.2 Learning Through Mutual Theory Refinement

Mutual theory refinement (MTR) (Kedar *et. al.*, 1991) is a multi-strategy learning method[3] used to refine the world models of a reactive system. Since these world models are always approximate in some way, occasionally the system may predict that a particular outcome of its actions will be observed, and instead encounter a *prediction failure*, where its predicted and observed states are at odds. Such failures drive the learning system to correct the world model so that future similar predictions are correct. MTR builds on earlier work in learning from failure (Schank, 1982; Simmons,1988; Sussman, 1985)and explanation-based learning (EBL) from failure for refining approximate theories (e.g. (Hammond, 1986; Chien, 1989)). MTR differs from most in that instead of assuming that a complete and correct theory is available to fix

[3]The term *multi-strategy learning* means that an array of learning techniques is available to apply to a problem, rather than a single method.

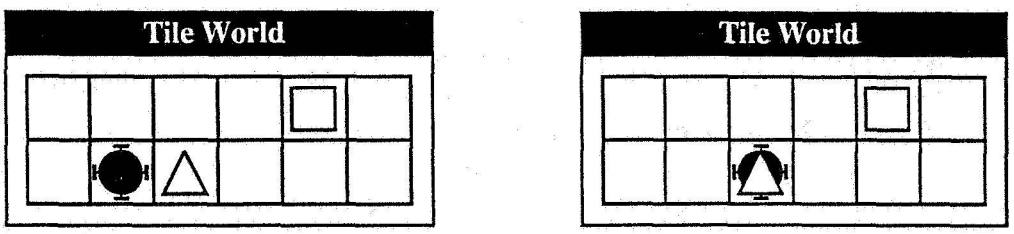

Figure 1: Missing Preconditions: observed and predicted states

the approximate one, MTR can refine one approximate theory with another, when possible. Otherwise, it degrades gracefully, using induction or rote learning.

The high-level algorithm for the control structure of MTR is as follows: MTR detects a prediction failure (different predicted and observed states). MTR then collects all possible diagnoses of the failure (existence or absence of literals in the predicted and observed states help determine what could have gone wrong). To repair the failure, a heuristic selection method selects the most effective repair. Each faulty literal in the prediction failure votes on one repair method. The method with the most votes gets used. To repair the failure, MTR searches through the set of available repair methods to check if one exists for this failure. If so, the failure is repaired. If not, other repairs are attempted. The system loops until all failures are repaired, or no more repair methods are available for this failure. In that case, the system updates the statistics collected on the probabilities of the variant outcomes, and outputs the refined operator or domain constraint model.

Prediction failures for ERE may result from missing or extra preconditions; missing or extra variant outcomes; incorrect preconditions or outcomes; missing, extra, or incorrect domain constraints. For purposes of this paper we present algorithmic descriptions for learning missing preconditions and missing variant outcomes only (see (Kedar *et. al*, 1991) for more detail). MTR performs plausible refinements which are annotated and may be revised with further information.

Learning missing preconditions: For ease of exposition, we focus on a simple NASA TileWorld example in Figure 1. (The agent is the black circle, and the tiles are the outlined figures.) Consider the MOVE operator. Suppose that the precondition that tests whether the destination cell is *clear* is missing

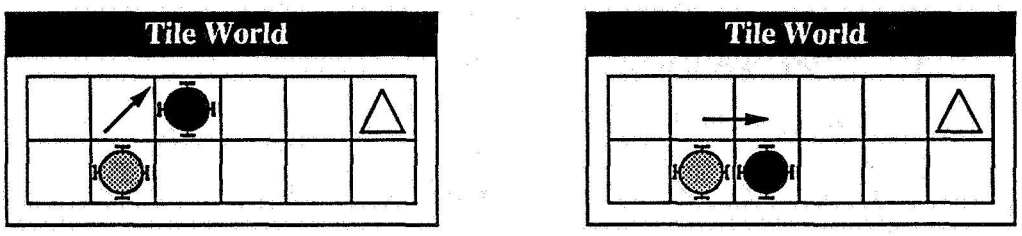

Figure 2: Missing variant outcomes: observed and predicted states

(similar errors of omission actually occurred as we wrote the operators for the original system). The reactor detects a prediction failure after it attempts unsuccessfully to move the agent to a cell containing a tile in it (an obstacle). The predicted outcome, that the agent be at the destination cell, differs from the actual outcome, in which the agent stays put because of the obstacle. Given as input the predicted and actual states, MTR explains the prediction failure as a violation of the domain constraint "two things cannot occupy the same cell." Using an algorithm similar to explanation-based learning from failure, the domain constraint is negated and regressed through the operator. It is added as a new (and generalized) precondition on the MOVE operator. It states that MOVE is applicable only if as a result "two things will not occupy the same cell" (or, before the move, the adjacent cell is *clear*)[4].

Learning variant outcomes: Again, consider the simple example above. For the MOVE operator, the only expected outcome is that the agent move straight to the destination cell. However, since the world (or in our case, the simulator) is not so predictable, the agent may veer to the right (or left) of its destination, illustrated as in Figure 2. (The agent's previous location is shown as a grey circle.) In such cases, the reactor will again detect a prediction failure. Given as input the predicted and actual states, the approximate domain constraint theory cannot explain the failure (unlike the above case, no domain constraint is violated by the prediction that the agent move straight). Without a theory, the learning method gracefully degrades to a weaker form of generalization (induction or rote learning). In this example,

[4]Note that this scenario illustrates that one approximate theory (the domain constraints) was sufficient to be used by MTR in refining another approximate theory (the operators). Conversely MTR can also mutually refine the approximate model of domain constraints using the approximate operator model.

it simply augments the operator with the observed outcome as an additional variant outcome, that of veering to the right (or left) of the destination cell.

3 Experimentation

While the goal of mutual theory refinement is to reduce prediction failures, its ultimate aim is to improve the overall performance of the associated system. In the following experiments, we illustrate how refinement by MTR affects ERE performance in a particular case study. To simplify the preliminary experiments, we focus initially on only two classes of refinements: adding missing preconditions and adding missing variant outcomes. The data was obtained by running the ERE/MTR system written in Allegro CommonLISP on a SUN SPARCstation 1 with 16M of memory. In order to facilitate the experimentation and avoid inefficiencies in our original implementation, we reimplemented simplified fast versions of MTR and ERE (as in Drummond & Levinson, 1991), simulating only the parts of the original systems needed for the experimentation. For the purpose of reproducing the experiments, the implementation and testbed can be obtained directly from the authors.

In these experiments, we provide the system with an *initial theory*: a faulty domain theory with no preconditions and variant outcomes, and observe the resulting effects on performance as the system corrects the theory. The system adds discovered missing preconditions and variant outcomes, resulting in an *endpoint theory*[5]. In effect, the domain theory is an independent parameter which changes over time as it is corrected and augmented. System performance is measured in terms of dependent parameters *accuracy* and *efficiency*. Accuracy is percent goal achievement, measured as the number of goals achieved by the integrated ERE planner and reactor. That is, it describes the success rate achieved by the reactor using the planner's advice (SCR's). Efficiency is measured in two ways. One is *search cost*, defined as the total number of nodes expanded during planning search. The second is *match cost*, defined as the number of function calls within the matcher when an operator is being tested for applicability.

Some things remained constant throughout the experiments. For all experiments we use 10 sets of 50 training problems and 100 test problems generated from the NASA TileWorld domain. All results are averaged over 10 runs. We fix the TileWorld grid size at 10 x 10 squares, with average

[5]By *endpoint theory* we mean a theory, still approximate, containing all preconditions and outcomes that have been discovered for these operators via learning.

tile density of 20%. Tile density in the TileWorld grid refers to the percentage of occupied grid cells. A density of 20% provides a scenario where planning significantly improves the system's goal achievement over a pure reactive strategy (Drummond, M., personal communication). Start and goal locations of the agent are randomly selected from the four corner positions of the grid, but are always situated diagonally across from one another (to provide problems of similar path length and difficulty).

Measuring accuracy while learning operator preconditions: The first experiment evaluates the effect of adding missing preconditions on accurate performance of the integrated ERE system. We begin with an initial faulty theory that is missing the precondition which tests 'obstacle in destination location?' in all four MOVE operator specializations. Each operator specialization represents whether the moving agent is hampered by tiles or the grid boundary on one or both sides, or is moving freely as in the center of the grid. To assess performance we measured the system's level of goal achievement as these preconditions were learned.

In Figure 3 we show the effects on system accuracy of adding missing preconditions. The horizontal axis displays the number of training instances, while the vertical axis displays the percent goal achievement attained by ERE. The important result to notice is that as MTR adds preconditions, more goals can be achieved by the combined planner and reactor in ERE. (The percent goal achievement asymptotes at 85% because goals are blocked by tiles in 15% of these problems, and therefore are impossible to achieve.) The key idea is that as long as the planner gives a plan to the reactor that is not faulty, the reactor can just step along the path the planner has suggested and achieve the goal. However, if the planner is using operators with missing preconditions (e.g. it is unaware of obstacles), a faulty plan may wrongly lead the reactor into an obstacle. When this happens the system registers a prediction failure, and the reactor halts without achieving the goal.

Measuring accuracy while learning variant outcomes: The second experiment evaluates the effect that adding variant outcomes has on accurate performance. The initial faulty theory contains 'moving straight' as the only outcome of the MOVE operators. However, we introduce 'effector errors' in the simulator, which allow the agent to veer to the right (10% of the time) or left (another 10% of the time) of a direction as it executes a move in that direction, making the single expected outcome, moving straight, inadequate for planning. To assess performance we measure the system's level of goal

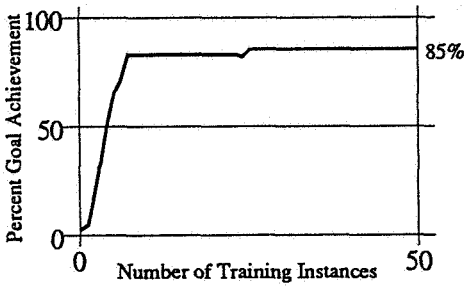

Figure 3. Learning curve for integrated system with incomplete initial domain theory. Training adds operator preconditions, improving overall goal achievement.

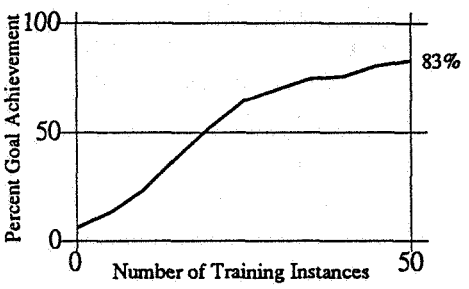

Figure 4. Learning curve with initial domain theory lacking variant outcomes. Training adds operator effects, improving overall goal achievement.

achievement as it learns the other missing variant outcomes.

We show the effects of adding missing variant outcomes on system accuracy in Figure 4. The important result to note is that as MTR adds variant outcomes (describing expected veers to the right or left) during training, the percentage of goals achieved on the test problems increases. After 50 training instances the asymptote of 85% is nearly reached. The key insight is that as the planner is equipped with a more complete operator model it can prepare more alternative plans for the reactor to get to the goal should it get off the main path. It uses a 'robustify' routine which considers high probability deviations from the path found by the planner, and creates additional SCR's to guide the reactor back to the path or straight to the goal, should it deviate from the main path (Drummond & Bresina, 1990). However, if the planner is missing variant outcomes (i.e. the planner is unaware of possible veers by the reactor) the planner may not prepare a 'contingency plan' for the reactor (how to get to the goal if it veers off the path into an unpredicted location). Thus when an unexpected veer happens, the system registers a prediction failure and the reactor halts without achieving the goal.

Measuring efficiency while learning preconditions: Figure 5 illustrates how adding preconditions affects the efficiency in terms of *match cost.* Match cost is measured in terms of the total number of function calls within the matcher as the planner finds applicable operators and constructs a plan. We show cost as a percentage of the maximum match cost measured with the

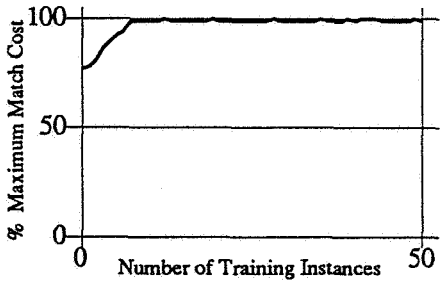

Figure 5. Cost of learning missing preconditions. Training results in increased match cost during planning and degraded system efficiency.

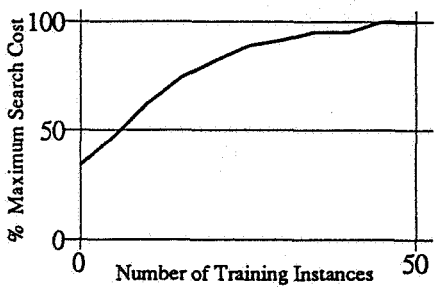

Figure 6. Cost of learning variant outcomes. Training results in increased node expansion during planning and degraded system efficiency.

fully acquired endpoint theory[6]. The curve does not begin at zero because the MOVE operators, even in the initial faulty theory, do contain some preconditions which are matched. Note that efficiency degrades as the system learns missing preconditions. As learning proceeds, the planner and reactor must gradually test more preconditions to determine whether an operator is applicable, hence match cost increases and efficiency degrades.

Measuring efficiency while learning variant outcomes: Figure 6 illustrates how adding variant outcomes affects the search cost. Search cost is measured in terms of the total number of nodes expanded (grid locations searched) during the planner's search for a solution. We show cost as a percentage of the maximum search cost (expanding every node on the grid, i.e. 100 nodes per problem). Important to note here is that as learning adds variant outcomes, the planner must explore more alternative paths to the goal during robustification, increasing search cost. To achieve optimal accuracy it searches the entire grid to handle low-probability veers, growing as $(O(n^2))$ with grid size.

4 Schemes for Computing Tradeoffs

Above we described experiments that demonstrated how MTR increased the accuracy of the associated ERE performance system, while degrading its effi-

[6]790 function calls per problem is the cost of testing all preconditions of the fully acquired endpoint theory for an average path length.

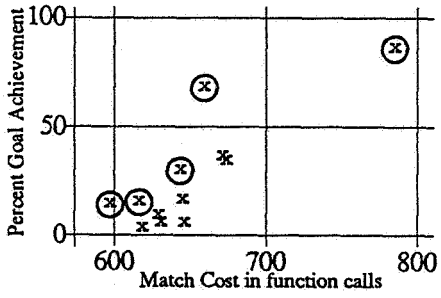

Figure 7. Learning curve for integrated system with incomplete initial domain theory. Training adds operator preconditions, improving overall goal achievement.

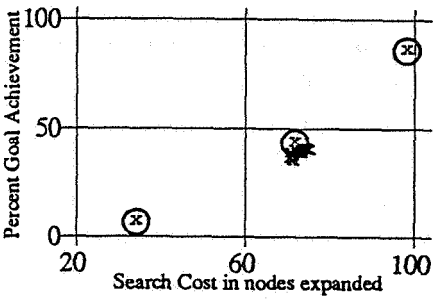

Figure 8. Learning curve with initial domain theory lacking variant outcomes. Training adds operator effects, improving overall goal achievement.

ciency. In this section we step away from the detailed experimental results of our case study in order to ascertain what general lessons we can draw about multi-strategy learning in integrated systems. Ultimately, multi-strategy learning in an integrated system needs to satisfy some global performance objectives. The difficulty, as we have shown in our specific experiments, is that there will often be multiple, possibly conflicting utility measures (as with accuracy versus efficiency in our case study). We need to investigate how to trade off *conflicting* utility metrics in order to achieve some global performance objectives.

Our preliminary solution is to provide data that clearly presents a range of possible tradeoffs. Then, given a particular performance objective, a tradeoff in conflicting utility measures can be selected that would satisfy the global objectives. In most previous work on the utility of learned knowledge, a cost-benefit tradeoff is calculated given a *single* dimension of utility (e.g. efficiency related to analytic learning in Minton, 1988; or accuracy related to empirical learning in Holder, 1991). Yet the cost-benefit tradeoff may appear in terms of different, and possibly conflicting dimensions of utility. We were inspired by (Ellman, 1988), who measures cost-benefit tradeoffs for approximated theories given conflicting utility metrics[7]. By measuring accuracy and efficiency at each point that a theory is approximated, a scatter plot graphing the data can

[7]Ellman's goal in measuring tradeoffs differs from ours; his aim is to determine how to trade off theory accuracy for tractability of explanation in EBL.

be produced to illustrate tradeoffs. On the scatter plot *pareto-optimal* points (boundary points) can be identified. A pareto-optimal point represents a version of the learned knowledge that cannot be improved in one dimension without degradation in the other dimension. To choose which of the pareto-optimal points is the desired tradeoff, one needs to determine how well the associated metrics satisfy some desired global performance objectives.

Improving efficiency by approximating preconditions: We illustrate this process using data from our case study. Once our system has fully acquired an endpoint theory, we introduce two methods for approximating this theory. By approximating the endpoint theory, we can increase system efficiency, and if the approximation is effective, can minimize the impact on accuracy. This can be done without destructively modifying the original body of learned knowledge. Our method differs from (Minton, 1988; Chase et al., 1989; and Cohen, 1990) who store knowledge in approximated form, as we seek to make dynamic approximations from an unabridged knowledge base.

For example, to improve efficiency in match cost once missing preconditions have been learned, we approximated certain preconditions by truifying or nullifying them (as in Keller, 1987). In particular, we set preconditions in various of the MOVE operators to 'almost always true' instead of actually matching them to check. Then we ran our sets of test problems on the system, measuring its resulting accuracy and efficiency using each of these approximated theories. The scatter plot in Figure 7 shows results for all the approximate theories we generated in this way (including the endpoint theory). The horizontal axis plots efficiency, as measured in match cost, while the vertical axis plots accuracy in terms of percent goal achievement. Each point on the graph represents the average tradeoff yielded by a particular approximated theory. Pareto-optimal points are circled.

Consider global objectives where desired accuracy is at least 60% goal achievement, with maximum possible efficiency gain. We find the pareto-optimal point which best satisfices the global objectives at 67% accuracy. This point in the scatter plot corresponds to an approximate operator model in which the precondition "clear-corridor" has been NILified. (Checking if a corridor is clear amounts to testing whether the destination cell is clear when surrounded by a 'corridor'–occupied cells on both left and right). Using this approximated theory we reduce our optimal accuracy of 85% by only 18% while reducing the match cost by 15% (from 780 to 660 per problem). Note

that the reason this particular approximation is effective is that for a grid with a 20% tile density, the chance of finding a 'corridor' configuration is not very great. Because the tradeoffs on these approximations have been explicitly plotted and measured, we are able to locate a meaningful approximation that identifies the optimal tradeoff in order to achieve our performance objectives.

Improving efficiency by approximating variant outcomes: As a second example, to improve efficiency in search once multiple variant outcomes have been learned, another method 'forgets', or approximates the operator model by pruning some of the learned variant outcomes. Recall that the independent variable dictating the 'effector errors' was set to 10% of the time for moving either right or left, and 80% of the time for moving straight. The scatter plot in Figure 8 shows results for approximating the variant outcomes of the move operator to just one or two outcomes (results for the endpoint theory containing all three outcomes are also shown). The horizontal axis plots efficiency, as measured in search cost, while the vertical axis plots accuracy in terms of percent goal achievement. Each point on the graph represents the average tradeoff yielded by a particular approximated theory.

The interesting pareto-optimal points on this graph are clustered at the center (all are optimal, plus or minus some statistical error). These points all represent approximated theories in which two most frequent out of the three variant outcomes are present (for each of the four operator specializations in four compass directions) resulting in sixteen approximate theories. These approximations decrease accuracy significantly (to 43% from 85%), while improving efficiency somewhat, decreasing the number of nodes expanded by nearly a third (from about 100 nodes per problem to about 72). The key insight is that when the model is approximated to constrain the variant outcomes used in planning to only the most frequently occurring ones, we decrease the planner's ability to prepare contingency paths should the agent veer in the unexpected way (hence the decrease in accuracy of goal achievement) but permit it to plan more efficiently (hence increasing efficiency).

In this graph, the best satisficing pareto-optimal point is rather disappointing, since we sacrifice quite a bit in accuracy to achieve minor gains in efficiency. A more sophisticated method might perform a 'percent probability approximation', only expanding variant outcomes greater than some threshold probability, and yield more useful tradeoff points. We conjecture that an optimal approximation could bring the worst-case search cost down

from polynomial ($O(n^2)$) to linear time (improving efficiency) while keeping goal achievement close to that attained when the entire grid is searched, thus maintaining accuracy.

Future work: The goal of these approximations is improved efficiency without much degraded accuracy, thus satisfying some global performance objectives. Given different objectives, different *dynamic* approximations on the same learned knowledge might be done to satisfice those. We are currently implementing an ERE/MTR performance system monitor that could gear the performance system to dynamically approximate the learned knowledge, sensitive to various performance measures and performance system components. Such an approach could lead to a flexible system which achieves goals efficiently without being required to limit or destructively modify its store of learned knowledge.

Acknowledgements

Lonnie Chrisman implemented version 2 of MTR. Thanks to John Bresina, Lonnie Chrisman, Mark Drummond, Pat Langley, Amy Lansky, Rich Levinson, Steve Minton, and Keith Swanson for helpful discussions regarding experimentation, and/or comments on earlier drafts.

References

Al-Badr, B. and Hanks, S. (1991). Critiquing the tileworld: Agent architectures, planning, benchmarks, and experimental methodology. Tech. Report University of Washington, FR-35, Seattle WA.

Berger, J. O. (1980). Statistical decision theory and bayesian analyses. New York: Springer-Verlag.

Bresina, J. and Drummond, M. (1990). Integrating planning and reaction: A preliminary report. *AAAI Spring Symposium Series.*

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2, 14-23.

Chien, S. A. (1989). Using and refining simplifications: Explanation-based learning of plans in intractable domains. *Proceedings of the Eleventh IJCAI*, Detroit, MI.

Drummond, M. and Bresina, J. (1990). Anytime synthetic projection: maximizing the probability of goal satisfaction. *Proceedings of the Ninth AAAI Conference* , Boston, MA.

Drummond, M., Bresina, J., and Kedar, S. (1991). The entropy reduction engine: integrating planning, scheduling, and control. *AAAI Spring Symposium Series.*

Drummond, M., and Levinson, R. (1991). How planning can help a reactive system. *Submitted to the First International Conference on AI Planning Systems.*

Ellman, T. (1988). Approximate theory formation: An explanation-based approach. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 570–574). St. Paul, MN: AAAI Press.

Hammond, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. *Proceedings of the Fifth AAAI Conference*, Philadelphia, PA.

Holder, L. B. (1988). *Maintaining the utility of learned knowledge using model-based adaptive control.* Master's thesis, Department of Computer Science, University of Illinois, Urbana, IL.

Holder, L. B. (1991). Selection of learning methods using an adaptive model of knowledge utility. *Proceedings of the First International Workshop on Multistrategy Learning.* (pp. 247–254). Princeton, NJ.

Kedar, S. and Bresina, J., and Dent, L. (1991). The blind leading the blind: Mutual refinement of approximate theories. *International Workshop in Machine Learning*, Evanston, IL, June, 1991.

Kedar, S. T. and Chrisman, L. (1992). Mutual theory refinement: A case study. Tech. Report NASA TR FIA-92-03, NASA Ames, June 1992.

Keller, R. (1987). Defining operationality of explanation-based learning. *Proceedings of the Sixth National Conference on Artificial Intelligence 2,* (pp. 482–487). Seattle, WA: AAAI Press.

Langley, P. and Drummond, M. (1990). Toward an experimental science of planning. *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 109–114). San Diego, CA: Morgan Kaufmann.

Langley, P. and Kibler, D. (1991). The experimental study of machine learning. Unpublished Manuscript.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence, 42,* 362–392.

Mooney, R. (1989). The effect of rule use on the utility of explanation-based learning. *Proceedings of the International Joint Conference on AI (IJCAI),* Detroit, MI (pp. 725–730), San Mateo: Morgan Kaufmann.

Philips, A. B. and Bresina, J. L. (1991). NASA TileWorld manual. Tech. Report NASA TR-FIA-91-04, NASA Ames. February, 1991.

Pollack, M. and Ringuette, M. (1990). Introducing the tileworld: Experimentally evaluating agent architectures. *Proceedings of the Eighth National Conference on Artificial Intelligence 2*, 183–189.

Schank, R. (1982). Dynamic memory: A theory of learning in computers and people. Cambridge University Press.

Simmons, R. (1988). A theory of debugging plans and interpretations. *Proceedings of the Seventh AAAI Conference*, Minneapolis, MN.

Simon, H. (1983). Why should machines learn? In R. S. Michalski, J. G. Carbonnell, and T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2.). San Mateo, CA: Morgan Kaufmann.

Sussman, G. (1975). A computer model of skill acquisition. New York: American Elsevier.

Tambe, M. and Rosenbloom, P. (1990). A framework for investigating production system formulations. *Proceedings of the Ninth AAAI Conference*, Boston, MA.